CPE/CSC 142

Term Project

Phase 2

Tyler Moua: 33% Contribution

Aaron Rai: 33% Contribution

Micaela Varquez: 33% Contribution

Table of Contents

Chapter Page

Status Report……………………...………………………….……………………………………4

Control Logic Representation……………………………………………………………………..7

ALUControlUnit…………………………………….…………………………………….7

BCHazardControlUnit…………………………………….………………………………8

ControlUnit…………………………………….………………………………………...10

RegisterForwardingUnit…………………………………….…………………………...12

Verilog Source Code……………………………………………………………………………..13

ALUControlUnit…………………………………………………………………………13

BCHazardControlUnit…………………………………….

BranchEquator…………………………………….

ControlUnit…………………………………….

CPU…………………………………….

DataMemory…………………………………….

EXMEM…………………………………….

IDEX…………………………………….

IFID…………………………………….

InstructionMemory…………………………………….

MainALU…………………………………….

MEMWB…………………………………….

MUX1-7…………………………………….

PC…………………………………….

RegisterFile…………………………………….

RegisterForwardingUnit…………………………………….

ShiftLeft…………………………………….

SignExtentions…………………………………….

ZeroExtend…………………………………….

Stimulus Module Used...…………………………………………………………………………

Datapath Diagram …………………………...……………………………………………………

***CSc/CPE 142***

***Term Project Status Report***

***Team #13***

Complete this form by typing the requested information and include the completed form in your report after TOC. Gray cells will be filled by the instructor.

|  |  |  |
| --- | --- | --- |
| *Name* | *% Contribution* | *Grade* |
| *Tyler Moua* | *33* |  |
| *Aaron Rai* | *33* |  |
| *Micaela Varquez* | *33* |  |

***Please do not write in the first table***

|  |  |
| --- | --- |
| *Project Report/Presentation 20%* | /200 |
| *Functionality of the individual components 40%* | /400 |
| *Functionality of the overall design 25%* | /250 |
| *Design Approach 5%* | /50 |
| Total points | /900 |

**A: List all the instructions that were implemented correctly and verified by the assembly program on your system:**

| Instructions | Was this instruction fully functional as verified by the assembly program provided? If no, explain. This refers to validation using the complete CPU and not its components. |
| --- | --- |
| Signed addition | Yes |
| Signed subtraction | Yes |
| Move | Yes |
| SWAP | Yes |
| AND immediate | Yes |
| OR immediate | Yes |
| Load byte unsigned | Yes |
| Store byte | Yes |
| Load | Yes |
| Store | Yes |
| Branch on less than | Yes |
| Branch on greater than | Yes |
| Branch on equal | Yes |
| jump | Yes |
| halt | Yes |

**B: Fill out the next table:**

| Individual Components | Does your system have this component? | List the student who designed and verified the block | Does it work? | List problems with the component, if any. |
| --- | --- | --- | --- | --- |
| ALU | Yes | Tyler | Yes |  |
| ALU control unit | Yes | Tyler | Yes |  |
| Memory Unit | Yes | Aaron | Yes |  |
| Register File | Yes | Aaron | Yes |  |
| PC | Yes | Micaela | Yes |  |
| IR | Yes | Micaela | Yes |  |
| Other registers | No |  |  |  |
| Multiplexors | Yes | Micaela | Yes |  |
| exception handler  1. Unknown opcode  2. Arith. Overflow  ….. | Yes | Tyler  Aaron  Micaela | Yes |  |
| Control Units   1. main 2. forwarding 3. lw hazard detection | Yes | Tyler  Aaron  Micaela | Yes | Flush occurs 1 stage later than expected. I.e. the first “stage to be flushed” is only flushed one cycle after a flush has been detected. |

How many stages do you have in your pipeline? 5

C: **State any issue regarding the overall operation of the datapath? Be Specific.**

Control Logic Representation

ALU Control Unit:

The ALU Control Unit sends an ALUControl signal to the Main ALU to communicate the arithmetic needed for the instruction in EX. The inputs include the ALUOP signal sent by the Control Unit as well as the Function Code of the instruction in the WB stage of the pipeline. The following is a truth table describing the output of the ALU Control unit based on various input values.

|  |  |  |
| --- | --- | --- |
| **Input** | | **Output** |
| ALUOP | FunctionCode | ALUControl |
| 0001 (A-Type) | 0000 (ADD) | 000 |
| 0001 (SUB) | 001 |
| 1110 (MOVE) | 010 |
| 1111 (SWAP) | 011 |
| 1001 (AND) | n/a | 100 |
| 1010 (OR) | n/a | 101 |
| 0100 (Load Byte Unsigned) | n/a | 000 |
| 0101 (Store Byte) | n/a | 000 |
| 0110 (Load) | n/a | 000 |
| 0111 (Store) | n/a | 000 |
| 1100 (Branch Less Than) | n/a | 000 |
| 1101 (Branch Greater Than) | n/a | 000 |
| 1110 (Branch Equal) | n/a | 000 |
| 0010 (Jump) | n/a | 000 |

BC Hazard Control Unit:

The BC Hazard Control Unit is the component that checks the datapath for hazard that require a bubble to be added in the pipeline. The inputs for this unit include: the opcodes for each instruction within the pipeline (ID, EX, MEM, WB) as well as the Hazard signal. The only output of this unit is the StopPC Signal.

The following a truth table for the BC Hazard Control Unit containing input values that enable the StopPC signal. Input values that not recorded in the table will result in a StopPC value of 0.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Input** | | | | **Output** |
| IDOP | EXOP | MEMOP | Hazard | StopPC |
| 0001 (A-Type) | 0110 OR 0100 (LW or LBU) | n/a | 1 | 1 |
| 1100 (BLT)  OR  1101 (BGT)  OR  1110 (BE) | 0001  (A-Type) | n/a | 1 | 1 |
| 0110 OR 0100 (LW or LBU) | n/a | 1 | 1 |
| n/a | 0110 OR 0100 (LW or LBU) | 1 | 1 |

Branch Equator:

The Branch Equator is the component that checks the values in Operand 1 and R15 to determine if a branch is valid. The inputs for this unit include the values of Operand 1 and R15 in ID, Back to Back and One Away forwarding Values, the BranchSelect, and HazardSelect signals, as well as the Hazard, Branch, and Jump signals. The only output of this component is the BranchingSoFlush Signal.

There is an internal reg “Operand” whose value is determined by the hazard and hazard select signals. The following is a truth table for determination of the value.

|  |  |  |
| --- | --- | --- |
| **Input** | | **Reg** |
| Hazard | Hazard Select | Operand |
| 0 | n/a | Op1 |
| 1 | 001 | BTB[15:0] |
| 1 | 010 | BTB[31:16] |
| 1 | 011 | OneAway[15:0] |
| 1 | 100 | OneAway[31:16] |

The following a truth table for the Branch Equator component, containing logic that enables the BranchingSoFlush signal. Input values that not recorded in the table will result in a StopPC value of 0.

|  |  |  |
| --- | --- | --- |
| **Input** | **Logic** | **Output** |
| Jump |  | BranchingSoFLush |
| 0 | Operand < R15 | 1 |
| 0 | Operand!<R15 | 1 |
| 0 | Operand==R15 | 1 |
| 1 | n/a | 1 |

Control Unit:

The Control Unit sends signals to many components of the datapath throughout each cycle of the pipeline. The inputs for this unit include: the opcodes for each instruction within the pipeline (ID, EX, MEM, WB), the function code of the instruction in WB, as well as the overflow signal. Additionally, this unit checks the opcode within ID to ensure that only predefined opcode values are used. Else, the halt signal will be enabled, stopping the pipeline.

The following are truth tables based on an instruction within a given stage of the pipeline. Note that these are processed concurrently, i.e. the Control Unit sends the signals for each instruction within their respective stages at the same time.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Input** | **Output** | | | |
| OpcodeID | OffsetSelect | Branch | BranchSelect | Jump |
| 0001 (A-Type) | 00 | 0 | 0 | 0 |
| 1001 (AND) | 01 | 0 | 0 | 0 |
| 1010 (OR) | 01 | 0 | 0 | 0 |
| 0100 (Load Byte Unsigned) | 00 | 0 | 0 | 0 |
| 0101 (Store Byte) | 00 | 0 | 0 | 0 |
| 0110 (Load) | 00 | 0 | 0 | 0 |
| 0111 (Store) | 00 | 0 | 0 | 0 |
| 1100 (Branch Less Than) | 01 | 01 | 00 | 0 |
| 1101 (Branch Greater Than) | 01 | 01 | 01 | 0 |
| 1110 (Branch Equal) | 01 | 01 | 10 | 0 |
| 0010 (Jump) | 10 | n/a | n/a | 1 |

|  |  |  |  |
| --- | --- | --- | --- |
| **Input** | **Output** | | |
| OpcodeEX | ALUOP | ALUSRC1 | ALUSRC2 |
| 0001 (A-Type) | OpcodeEX | 00 | 0 |
| 1001 (AND) | OpcodeEX | 01 | 0 |
| 1010 (OR) | OpcodeEX | 01 | 0 |
| 0100 (Load Byte Unsigned) | OpcodeEX | 00 | 1 |
| 0101 (Store Byte) | OpcodeEX | 00 | 1 |
| 0110 (Load) | OpcodeEX | 00 | 1 |
| 0111 (Store) | OpcodeEX | 00 | 1 |
| 1100 (Branch Less Than) | OpcodeEX | 10 | 0 |
| 1101 (Branch Greater Than) | OpcodeEX | 10 | 0 |
| 1110 (Branch Equal) | OpcodeEX | 10 | 0 |
| 0010 (Jump) | xxxx | 0 | 0 |

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | |
| OpcodeMEM | MemRead | StoreOffset |
| 0001 (A-Type) | 0 | 0 |
| 1001 (AND) | 0 | 0 |
| 1010 (OR) | 0 | 0 |
| 0100 (Load Byte Unsigned) | 1 | 0 |
| 0101 (Store Byte) | 1 | 1 |
| 0110 (Load) | 1 | 0 |
| 0111 (Store) | 1 | 0 |
| 1100 (Branch Less Than) | 0 | 0 |
| 1101 (Branch Greater Than) | 0 | 0 |
| 1110 (Branch Equal) | 0 | 0 |
| 0010 (Jump) | 0 | 0 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Input** | | **Output** | | |
| OpcodeEX | FunctionCode | MemToReg | RegWrite | WriteOp2 |
| 0001 (A-Type) | 1111 (Swap) | 00 | 1 | 1 |
| !(1111) (Not Swap) | 00 | 1 | 0 |
| 1001 (AND) | n/a | 00 | 1 | 0 |
| 1010 (OR) | n/a | 00 | 1 | 0 |
| 0100 (Load Byte Unsigned) | n/a | 01 | 1 | 0 |
| 0101 (Store Byte) | n/a | 0 | 0 | 0 |
| 0110 (Load) | n/a | 01 | 1 | 0 |
| 0111 (Store) | n/a | 0 | 0 | 0 |
| 1100 (Branch Less Than) | n/a | 0 | 0 | 0 |
| 1101 (Branch Greater Than) | n/a | 0 | 0 | 0 |
| 1110 (Branch Equal) | n/a | 0 | 0 | 0 |
| 0010 (Jump) | n/a | 0 | 0 | 0 |

Register Forwarding Unit:

The Register Forwarding Unit is a component that detects hazards based on the operands of various instructions within the pipeline and outputs the hazard signal as well as various MUX selection signals. The inputs of this unit include: The first operand of the instruction in ID as well as both operands of each instruction within the other stages of the pipeline (EX, MEM, WB).

The following are truth tables for the Register Forwarding Unit containing input values that enable the HazardDetected signal. Input values that not recorded in the table will result in a HazardDetected value of 0. Additionally, each truth table in broken between different MUX selection signals. When there is no hazard detected, the selection signal for the MUXes are set to 0.

NOTE: Mux 4 was removed from the final pipeline, but the signal “ForwardToMux4” is used by the BranchEquator component for forwarding.

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | |
| EXOP2 | ForwardToMux3 | HazardDetected |
| MEMOP1 | 001 | 1 |
| WBOP1 | 011 | 1 |
| MEMOP2 | 010 | 1 |
| WBOP2 | 100 | 1 |

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | |
| IDOP | ForwardToMux4 | HazardDetected |
| EXOP1 | 001 | 1 |
| MEMOP1 | 011 | 1 |
| EXOP2 | 010 | 1 |
| MEMOP2 | 100 | 1 |

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | |
| EXOP1 | ForwardToMux5 | HazardDetected |
| MEMOP1 | 001 | 1 |
| WBOP1 | 011 | 1 |
| MEMOP2 | 010 | 1 |
| WBOP2 | 100 | 1 |

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | |
| BTBOP1 | ForwardToMux6 | HazardDetected |
| OAOP1 | 001 | 1 |

Verilog Source Code

Stimulus Module Used

Datapath Diagram